Lab No. 3
Time-Series Stock Forecasting with RNNs/LSTMs/GRUs

1. Learning Objectives

- Apply recurrent neural architectures (vanilla RNN, LSTM, GRU) to univariate and multivariate financial time-series forecasting.
- Engineer and evaluate sliding-window datasets (lookback/forecast horizons) for supervised sequence modeling.
- Implement training/evaluation loops in PyTorch with proper scaling, batching, early stopping, and checkpointing.
- Compare models, hyperparameters, and validation strategies using strong statistical and visual analyses.
- Benchmark against naïve baselines (last-value, simple moving average) and, optionally, classical models (ARIMA/Prophet).

2. Problem Statement

Select a publicly traded company (e.g., AAPL, MSFT, TSLA) and build a sequence model to forecast its closing price (or returns) over a short horizon (e.g., next 1–5 trading days). Your final notebook must demonstrate end-to-end work: data acquisition, preprocessing, modeling, training, evaluation, error analysis, and clear conclusions.

3. Dataset

Use one of the following sources to obtain daily historical data (minimum 5 years if available):

- Yahoo Finance via the `yfinance` Python package (OHLCV).
- Stooq (accessible through `pandas_datareader` or `yfinance` fallback).
- Optionally, Kaggle financial datasets (be sure to cite the exact source and include a data dictionary).

Required fields: Date, Open, High, Low, Close, Volume. You may derive additional features such as moving averages, RSI, MACD, Bollinger Bands, rolling volatility, sector ETF signals, or market index features (e.g., SPY, ^VIX). Ensure you document any feature engineering choices.

4. Modeling Requirements

- Implement at least **two** recurrent variants among: vanilla RNN, LSTM, GRU (PyTorch).
- Use a supervised sliding-window setup with a lookback window (e.g., 30–120 days) and a forecast horizon (e.g., 1–5 days).
- Train/validation/test split must be strictly chronological. Consider **rolling-origin** (walk-forward) validation to reduce temporal leakage.
- Implement at least **one** multivariate configuration (include technical indicators or market covariates) and **one** univariate configuration.
- Report metrics: MAE, RMSE (or MSE), and MAPE (with safeguards for near-zero denominators).
- Include a **naïve baseline** (last observed value) and a **moving-average baseline**; optionally include ARIMA/Prophet for reference.
- Use appropriate normalization (e.g., MinMax or StandardScaler) fit only on training data to avoid leakage.
- Early stopping on validation loss; save best model checkpoints; include a learning-rate scheduler (e.g., ReduceLROnPlateau).

5. Visualizations (Required)

- Line plots of raw Close prices with train/val/test spans shaded.
- Candlestick charts for selected periods (e.g., last 6–12 months) to contextualize predictions (use `mplfinance`).
- Feature/indicator overlays: moving averages, Bollinger Bands, RSI, etc.
- Training curves: loss and metric vs epochs for each model/setting.
- Prediction vs actual plots on the test period (include confidence/quantile bands or empirical residual spread, if computed).
- Residual analysis: residual time series, histogram, and autocorrelation (ACF) to detect structure left in errors.
- Walk-forward validation diagram: depict rolling windows and aggregated performance.

6. Recommended Notebook Structure

1. Introduction & Research Questions
2. Data Acquisition & Documentation (source, period, ticker, fields)
3. Preprocessing & Feature Engineering (scaling, windowing, leakage safeguards)
4. Baselines (naïve, moving average; optional: ARIMA/Prophet)
5. Model Architectures (RNN/LSTM/GRU) and Rationale
6. Training Protocols (splits, early stopping, scheduler, checkpoints)
7. Experiments & Ablations (hyperparameters, windows, horizons, features)
8. Results & Visualizations (metrics tables, curves, plots)
9. Error Analysis (residuals, failure cases, regime shifts)
10. Conclusions & Lessons Learned (limitations, future work)

11. Reproducibility Notes (random seeds, environment, versions)

7. Deliverables

- A single Jupyter notebook (`.ipynb`) with all code, outputs, and narrative text (Markdown).
- A short **executive summary** (≤ 1 page inside the notebook) explaining your best model, validation choice, and what worked/failed.
- Saved best model weights and a small utility function to reload and run on the last 60 days to produce next-step prediction(s).
- A `requirements.txt` (or `environment.yml`) to reproduce the environment; note PyTorch + CUDA versions where relevant.
- If using external datasets (e.g., Kaggle), include a README with exact dataset links and citation.

8. Implementation Guidelines

- Use `yfinance` to download OHLCV. Cache CSV locally to ensure reproducibility.
- Build a PyTorch `Dataset` for sliding windows; implement `__getitem__` to return (X_window, y_target).
- Start with a simple univariate LSTM predicting next-day close or return; then extend to multivariate inputs and multi-step outputs.
- Implement early stopping (patience ~10) on validation loss; save best weights using `torch.save`.
- Track experiments with a lightweight logger (e.g., CSV + matplotlib plots).
- Set `torch.manual_seed` and NumPy/Random seeds; document CUDA/cuDNN version.

9. Ethical & Practical Considerations

- This assignment is for educational purposes only; **not** financial advice.
- Markets are non-stationary; models may fail under regime shifts (e.g., crises, policy changes).
- Avoid data snooping and look-ahead biases; clearly mark all decisions made using only training data.
- Be transparent about limitations and uncertainty in predictions.

10. Submission Checklist

- Notebook runs top-to-bottom without errors and reproduces reported metrics/plots.
- All required visualizations present and labeled (axes, units, time frames).
- Baselines implemented and compared against your best model.
- Executive summary included; conclusions justified by evidence.
- Environment file and random seeds provided.